countEvens – contarPares

Retorne la cantidad de enteros positivos en el arreglo dado. Nota: el operador % “módulo” computa el residuo, ej.: 5%2 es 1.

bigDiff - granDifferencia

Dado un arreglo de enteros de longitud 1 o más, retorne la diferencia entre el valor más grande del arreglo y el valor más pequeño de este. Nota: las funciones de java Math.min(a, b) y Math.max(a, b) retornan el valor más pequeño o más grande, respectivamente, de los dos dados.

centeredAverage – promedioCentrado

Retorne el promedio “centrado” de un arreglo de enteros, el cual diremos que es la media aritmética de los valores (promedio), sólo que se ignorarán del conteo tanto el valor máximo del arreglo como el valor mínimo. Si hay múltiples copias del valor mínimo, ignore sólo una; Si hay múltiples copias del valor máximo, ignore sólo una. Use división entera para producir el promedio final. Asuma que la longitud del arreglo es mayor o igual a 3.

sum13 – suma13

Retorne la suma de los números en el arreglo de enteros dado, retornando 0 para un arreglo vacío. Excepto que el número 13 tiene mucha mala suerte, así que no cuenta en la suma y los números que aparecen inmediatamente después de este tampoco.

sum67 – suma67

Retorne la suma de los números en el arreglo de enteros dado, pero ignore las secciones de números que inician con 6 y se extienden hasta el siguiente 7 (todo 6 estará seguido de por lo menos un 7). Retorne 0 para un arreglo vacío.

has22 – tiene22

Dado un arreglo de enteros, retorne verdadero si el arreglo contiene un 2 luego de otro 2 en alguna parte de este.

lucky13 – suertudo13

Dado un arreglo de enteros, retorne verdadero si el arreglo no contiene ni 1s ni 3s.

sum28 – suma28

Dado un arreglo de enteros, retorne verdadero si la suma de todos los 2s en el arreglo es exactamente 8.

more14 – mas14

Dado un arreglo de enteros, retorne verdadero si la cantidad de 1s en el arreglo es mayor que la cantidad de 4s en el arreglo.

fizzArray – arregloEfervescente

Dado un número n, cree y retorne un nuevo arreglo de enteros de longitud n que contenga los números 0, 1, 2,…, n-1. El n dado puede ser 0, en cuyo caso simplemente retorne un arreglo de longitud 0. No es necesario que haga un if separado para el caso en que n = 0; el ciclo for debería ejecutarse 0 veces en este caso. La sintaxis para hacer un nuevo arreglo es: new int[tamaño_deseado].

only14 – solo14

Dado un arreglo de enteros, retorne verdadero si todos los elementos son 1 o 4.

fizzArray 2– arregloEfervescente2

Dado un número n, cree y retorne un nuevo arreglo de strings de longitud n que contenga los strings "0", "1", "2", …, hasta n-1. n puede ser 0, en cuyo caso simplemente retorne un arreglo de longitud 0. Nota: String.valueOf(xxx) convertirá xxx en un string y funciona para casi cualquier tipo de dato. La sintaxis para hacer un nuevo arreglo es: new int[tamaño_deseado].

no14 – no14

Dado un arreglo de enteros, retorne verdadero si este no contiene ni 1s ni 4s.

isEverywhere – estáEnTodosLados

Diremos que un valor "está en todos lados" en un arreglo si por cada par de elementos adyacentes en el arreglo (con overlapping o superposición), por lo menos uno de los elementos del par es ese valor. Retorne verdadero si el valor dado está "en todos lados" en el arreglo.

either24 – 2o4

Dado un arreglo de enteros, retorne verdadero si el arreglo contiene un 2 luego de otro 2 o un 4 luego de otro 4, pero no ambos.

matchUp – emparejamiento

Dado dos arreglos de enteros, nums1 y nums2 de la misma longitud, por cada elemento en nums1, considere el elemento correspondiente en nums2 (el elemento en el mismo índice). Retorne la cantidad de veces que los elementos difieren por una cantidad de 2 o menos, pero no son iguales. Nota: la función de Java Math.abs(a, b) calcula la diferencia entre ambos números.

has77 – tiene77

Dado un arreglo de enteros, retorne verdadero si el arreglo contiene dos 7s juntos, o hay dos 7s separados por un elemento, como por ejemplo en {7, 1, 7}.

has12 – tiene12

Dado un arreglo de enteros, retorne verdadero si hay un 1 en el arreglo con un 2 en algún lugar más tarde en este mismo.

modThree – moduloTres

Dado un arreglo de enteros, retorne verdadero si el arreglo contiene, juntos, ya sea 3 valores pares o 3 valores impares.

haveThree – tieneTres

Dado un arreglo de enteros, retorne verdadero si el valor 3 aparece en el arreglo exactamente 3 veces y ningún 3 está inmediatamente después de otro 3.

twoTwo – dosDos

Dado un arreglo de enteros, retorne verdadero si todo 2 que aparece en el arreglo está inmediatamente después de otro dos.

sameEnds – mismosFinales

Retorne verdadero si el grupo de N números al principio y al final del arreglo son los mismos. Por ejemplo, con {5, 6, 45, 99, 13, 5, 6} los finales son los mismos para n=0 y n=2, pero son diferentes para n=1 y n=3. Asuma que N está en el rango [0, nums.length]

tripleUp – aumentoTriple

Retorne verdadero si el arreglo de enteros dado contiene, en algún lugar, tres números adyacentes que se incrementan, de izquierda a derecha, en una unidad.

fizzArray3 - arregloEfervescente3

Dados dos números enteros, inicio (start) y fin (end), retorne un nuevo arreglo que contenga la secuencia de enteros desde el inicio pero sin incluir el final, así para start=5 y end=10 tenemos {5, 6, 7, 8, 9}. El número final será mayor o igual que el número incial. Note que un arreglo de longitud 0 es válido.

shiftLeft – desplazarIzquierda

Retorne un arreglo que está “desplazado a la izquierda” en una unidad. Ej.: para {6, 2, 5, 3} obtendríamos {2, 5, 3, 6}. Note que el primer número pasa al final. Puede modificar y retornar el arreglo dado, o puede retornar un nuevo arreglo.

tenRun – corridaDiez

Dado un arreglo de enteros, para cada múltiplo de 10 que encuentre en este, reemplace todos los números que sigan a este por el (último) múltiplo de 10 que encontró, hasta que encuentre un nuevo múltiplo de 10. Ej.: para {2, 10, 3, 4, 20, 5} obtendríamos {2, 10, 10, 10, 20, 20}.

pre4 – pre4

Dado un arreglo no vacío de enteros, retorne un nuevo arreglo que contenga los elementos del arreglo original que están antes del primer 4 en el arreglo original. El arreglo original contendrá por los menos un 4. Tenga en cuenta que en java es válido crear un arreglo de longitud 0.

post4 – post4

Dado un arreglo no-vacío de arreglos, retorne un nuevo arreglo de enteros que contenga los elementos del arreglo original que estén después del último 4 que aparezca en el arreglo. El arreglo original contendrá por lo menos un 4. Tenga en cuenta que en java es válido crear un arreglo de longitud 0.

notAlone – noSolo

Diremos que un elemento en un arreglo está "solo" si hay valores antes y después de este, y estos valores son diferentes de él. Retorne una versión del arreglo dado en donde cada instancia del valor dado que esté sola es reemplazada por el valor mayor a su izquierda o derecha.

zeroFront – frontalZero

Retorne un arreglo que con tiene exactamente los mismos números que el arreglo dado, pero reordenados de manera que todos los 0s están agrupados al inicio del arreglo. El orden de los números diferentes de 0 no importa. Ej.: {1, 0, 0, 1} se convierte en {0 ,0, 1, 1}. Puede modificar y retornar el arreglo dado, o puede retornar un nuevo arreglo.

withoutTen – sinDiez

Retorne una versión del arreglo dado en donde todos los 10 han sido removidos. Los elementos restantes deberán agruparse al inicio (conservando su orden), y los espacios vacíos que quedaron en el final deberán ser 0. Ej.: {1, 10, 10, 2} produciría {1, 2, 0, 0}. Puede modificar y retornar el arreglo dado, o puede retornar un nuevo arreglo.

zeroMax – maxCero

Retorne una versión del arreglo dado en donde cada 0 que hay en el arreglo es reemplazado por el valor impar más grande que hay (en el arreglo) a la derecha de este. Si no hay valores impares a la derecha del 0, deje el 0 como tal.

evenOdd – parImpar

Retorne un arreglo que contenga exactamente los mismos números que en el arreglo dado, pero reordenados de manera que todos los números pares están antes que los números impares. Luego de cumplir está condición, los números pueden estar en cualquier orden. Es decir, tiene que agrupar pares a la derecha e impares a la izquierda, pero el orden en que aparecen los elementos pertenecientes a cada grupo es irrelevante. Puede modificar y retornar el arreglo dado, o puede retornar un nuevo arreglo.

fizzBuzz – FizzBuzz (problema "famoso")

Esta es una versión un poco más difícil del famoso problema FizzBuzz que algunas veces es presentado como primer problema en entrevistas (técnicas) de trabajo. Considere la serie de números que empiezan en inicio (start) y que van hasta fin (end) sin incluir este último (exclusivo), de manera que, por ejemplo, para start=1 y end=5 tenemos la serie 1, 2, 3, 4. Retorne un nuevo arrego de strings (String[]) que contenga estos números pero en forma de cadena, excepto que para

múltiplos de 3 debe poner “Fizz”, para múltiplos de 5 debe poner “Buzz”, y para múltiplos de 3 y 5 al mismo tiempo debe poner “FizzBuzz”. En java el método String.valueOf(xxx) creará un string a partir de un entero u otro tipo. Esta versión es un poco más complicada que la versión usual ya que ud. tiene que guardar la respuesta en un arreglo (y por tanto, manejar índices) en vez de simplemente imprimir, y también porque variamos el inicio y el final en vez de utilizar siempre el rango [1, 100).